

`09 Summer SPARCS Seminar

Introduction to Python #2

SPARCS `08

서우석(pipoket)

► Last Time

You have learned...

How to Install and Run Python

How to Use Python as Calculator

Basic Python Syntax

(indentation, quote, multiline text...)

Python Data Structure

(string, number, list, dictionary, tuple...)

Basic Input, Output to the terminal

(input_raw, input, print)

Basic File Operation

(open, write, readline, close ...)

▶ Last Homework

- You should review all the materials we have discussed
- You can get the codes used in this ppt http://pipoket.kaist.ac.kr/sp_seminar/week01.tar.gz
- Following command will expand the file on Linux

```
~# tar -zxvf week01.tar.gz
```

► Last Homework

Your program should do the following

1. Open the given file diy6.db
2. File is like this

```
"20080421[TAB]Woosuk Suh[TAB]Computer Science"
```

```
"20080719[TAB]Chanhee Lee[TAB]Undecided"
```

3. Read the file and save the data as you wish
4. Get the Student ID as input and print out the information of student

▶ Brief Contents

Function

Class with OOP

(Inheritance, Overloading, Overriding ...)

Package

Advanced Python Topics

Simple Format String

List Generating

Generator (yield)

Lambda Function (lambda)

▶ Function



► Function

```
1 def function(arg1, arg2, arg3=default):  
2     # Do something with arg1, arg2  
3     # ... ..  
4     return result1, result2
```

► Function

```
1 #!/usr/bin/env python
2 # -*- coding: utf-8 -*-
3
4 # Lets' define some functions
5 def get_sum(first, second):
6     """Calculates sum of two given numbers"""
7     sum = first + second
8     return sum
9
10 def get_diff(first, second):
11     """Calculates difference of two given numbers"""
12     diff = first - second
13     if first < second:
14         diff *= -1
15     return diff
16
17 def get_multi(first=1, second=2):
18     """Calculates multiplication of two given numbers"""
19     mul = first * second
20     return mul
21
22 def get_div(dividend, divisor):
23     if divisor == 0:
24         return "Dividing by zero is not allowed!"
25     return dividend / divisor
```

► Function

```
1 #!/usr/bin/env python
2 # -*- coding: utf-8 -*-
3
4 # Lets' define some functions
5 def get_sum(first, second):
6     """Calculates sum of two given numbers"""
7     sum = first + second
8     return sum
9
10 def get_diff(first, second):
11     """Calculates difference of two given numbers"""
12     diff = first - second
13     if first < second:
14         diff *= -1
15     return diff
16
17 def get_multi(first=1, second=2):
18     """Calculates multiplication of two given numbers"""
19     mul = first * second
20     return mul
21
22 def get_div(dividend, divisor):
23     if divisor == 0:
24         return "Dividing by zero is not allowed"
25     return dividend / divisor
26
27 # Using function is like this
28 print get_sum(1, 2) # 3
29 print get_diff(2, 5) # 3
30
31 # If no input is given, default is used
32 print get_multi() # 2
33 # Else, the given input is used
34 print get_multi(4, 3) # 12
35
36 # You don't have to care about return type
37 print get_div(4, 0) # Text returned
38 print get_div(4, 2) # Integer returned
39
40 # You can specify input in detail
41 print get_multi(first=4, second=2) # 8
42 print get_multi(second=4) # 4
43
44 # But this is not allowed, specifying argument in detail
45 # must be used on the last arguments
46 # get_sum(first=4, 3) XXX
47 # get_sum(first=4, 3) XXX
```

▶ Class



CS101의 추억

클래스?

자동차 설계도

오브젝트?

생산된 자동차

▶ Class

OOP의 가장 중요한 개념!

Object :: 자료 + 처리 방법

Class :: Object의 설계도

굳이 비유 하자면

Object :: MP3 Player (파일 + 재생방법)

Class :: MP3 Player의 설계도



▶ Class

```
1 class MP3Player:
2     def __init__(self):
3         """Constructor"""
4         self.files = get_musics()
5         self.pictures = get_pictures()
6         self.datetime = get_datetime()
7         self.status = get_status()
8
9     def __del__(self):
10        """Destructor"""
11        save_datetime(self.datetime)
12        save_status(self.status)
13        shutdown()
14
15    def play(self, filename):
16        try:
17            self.status = "PLAY"
18            file = self.files[filename]
19            file.open()
20            self.start_decode(file)
21            return True
22        except Exception:
23            file.close()
24            self.status = "STOP"
25            return False
```



No public, private
기본적으로 모두 public

리턴타입이 없다
Dynamic Typing

self. self. self. self...
오브젝트 자신을 지칭
항상 붙여야 함

▶ Class

오브젝트 생성하는 방법

```
31 ipod_0001 = MP3Player()  
32 ipod_0002 = MP3Player()  
33  
34 ipod_0001.play("She`s gone.mp3")  
35 ipod_0002.play("[09년도 3주차][멜론 TOP50]소녀시대-힘내!.mp3")
```

오브젝트의 메소드 호출하는 방법

No "new ClassName()"
그냥 ClassName()



▶ Class – Constructor, Destructor

Constructor

Object가 생성될 때 실행
주로 Object를 초기화 하는 코드

Destructor

Object가 파괴 Garbage Collection 될 때 실행
주로 가져다 쓴 자원을 돌려주는 코드

▶ Class – Constructor, Destructor

꼭 선언할 필요 없음

기본 Constructor, Destructor 존재

Argument를 받을 수 있음

리턴 값이 있어서는 안됨

▶ Class – Constructor, Destructor

생성자 선언하는 방법

```
1 class MP3Player:
2     def __init__(self):
3         """Constructor"""
4         self.files = get_musics()
5         self.pictures = get_pictures()
6         self.datetime = get_datetime()
7         self.status =
8
9     def __del__(self):
10        """Destructor"""
11        save_datetime(self.datetime)
12        save_status(self.status)
13        shutdown()
```

파괴자 선언하는 방법

▶ Class – Member Variables

Object가 갖고 있는 자료

형식

```
self.variable_name
```

Object마다 다른 값을 가질 수 있음

MP3 Player마다 서로 다른 곡

MP3 Player마다 약간 다른 현재 시각

▶ Class – Member Variables

```
1 class MP3Player:
2     def __init__(self):
3         """Constructor"""
4         self.files = get_musics()
5         self.pictures = get_pictures()
6         self.datetime = get_datetime()
7         self.status = get_status()
8
9     def __del__(self):
10        """Destructor"""
11        save_datetime(self.datetime)
12        save_status(self.status)
13        shutdown()
```

선언하는 방법

가져다 쓰는 방법
Don't forget "self"!!

▶ Class – Inheritance, Overriding

모든 Player로 사업 확장!



생각해보니 play, stop은...
DVD Player, CD Player,
Video Player...

좀 편하게 쓸 방법 없나?

▶ Class – Inheritance, Overriding

Player의 공통된 기능들 모아 클래스
☞ (Abstract) Class

Player들은 이 클래스의 기능을 활용
☞ 상속 Inheritance

모자란/맞지 않는 기능은 알아서 변경
☞ 오버라이딩 Overriding

▶ Class – Inheritance

```
1 class GenericPlayer:
2     def __init__(self):
3         check_machine()
4         self.files = get_files()
5         self.datetime = get_datetime()
6         self.status = get_status()
7
8     def __del__(self):
9         save_datetime(self.datetime)
10        save_status(self.status)
11        shutdown()
12
13    def play(self, filename):
14        try:
15            self.status = "PLAY"
16            file = self.files[filename]
17            file.open()
18            self.start_decode(file)
19            return True
20        except Exception:
21            file.close()
22            return False
```

상속 받는 방법

```
25 class MP3Player(GenericPlayer):
26     def __init__(self):
27         """Constructor"""
28         GenericPlayer.__init__(self)
29         self.pictures = get_pictures()
```

상속 받은 클래스의
Constructor 호출

```
ipod = MP3Player()
ipod.play("Song.mp3")
```

▶ Class – Overriding

MP3 Player, DVD Player
파일 기반의 플레이어

CDPlayer
트랙 기반의 플레이어

... GenericPlayer는 무용지물?!

▶ Class – Overriding

```
31 class CDPlayer(GenericPlayer):
32     def __init__(self):
33         """Constructor"""
34         GenericPlayer.__init__(self)
35         self.cdrom = get_cdrom()
36         self.pictures = get_pictures()
37
38     def play(self, trackno):
39         """MusicCD must be played by track"""
40         try:
41             self.status = "PLAY"
42             track = self.cdrom.get_track(trackno)
43             self.cdrom.rotate()
44             self.start_read_track(track)
45             return True
46         except Exception:
47             self.cdrom.stop()
48             return False
```

이미 있지만, 맞지 않다!
그럼 새로 선언하면 된다

▶ Class? OOP!

정리!

Python의 특징

1. 큰 모듈을 만들기 쉽다

2. 동적 타입

3. 상속

4. 오버라이딩

지금까지 배운건...

Not only for Python!

But also for all OOP!

다만, 다시 다룬 이유는...

Python에서는 이렇게

Python에서는 이런 Syntax

어! 헛갈리면 안돼요!

▶ Package

거대한 프로젝트, 엄청난 코드
적당히 나누기
적당히 분배하기

나눈 파일을 가져다가 쓰는 방법!
Package 활용

▶ Package

```
mypackage.py  calc.py
1 class Calculator(object):
2     def __init__(self, first, second):
3         self.first = first
4         self.second = second
5
6     def get_sum(self):
7         return self.first + self.secon
```

이렇게 다른 파일로 빼고

```
mypackage.py  calc.py
1 import mypackage
2 calculator = mypackage.Calculator(
3     first=3, second=4)
4 print calculator.get_sum()
5
6
7 from mypackage import Calculator # Importing class
8 calculator = Calculator(first=5, second=6)
9 print calculator.get_sum()
```

이렇게 가져다가 쓴다

▶ Package

아하! 없는 건 에러나네

```
In [1]: import notexist
```

```
-----  
-----  
ImportError  
call last)
```

Traceback (most recent

```
/home/pipoket/project/sp_python_seminar/week2/<ipython console> i  
n <module>()
```

```
ImportError: No module named notexist
```

```
In [2]: import sys
```

```
In [3]: █
```

어? 저는 sys 만든 적 없는데요??

▶ Package

```
In [3]: sys.path
Out[3]:
['',
 '/usr/bin',
 '/usr/lib/python2.5/site-packages/GChartWrapper-0.8-py2.5.egg',
 '/usr/lib/python2.5',
 '/usr/lib/python2.5/plat-linux2',
 '/usr/lib/python2.5/lib-tk',
 '/usr/lib/python2.5/lib-dynload',
 '/usr/local/lib/python2.5/site-packages',
 '/usr/lib/python2.5/site-packages',
 '/usr/lib/python2.5/site-packages/PIL',
 '/var/lib/python-support/python2.5',
 '/var/lib/python-support/python2.5/gtk-2.0',
 '/var/lib/python-support/python2.5/IPython/Extensions',
 '/home/pipoket/.ipython']
```

기본 Python의 Package 경로

저 폴더 안의 것들은 자동으로 import 가능

▶ Package

/some/path

여기에도 `__init__.py`

`package1/__init__.py`

`package1/module1.py`

`package1/module2.py`

`package2/__init__.py`

`package2/module1.py`

`package2/module2.py`

`sys.path`에

`/some/path`만 있으면

`__init__.py` 덕분에

```
import path.package1.module1
```

```
import path.package2.module1
```

▶ Package

Do you remember?

Python의 장점

수 많은 Packages

여러분이 할 삽질을..

이미 누군가가 해 놨다!

그들의 Package를 활용하면

보다 윤택한 Python 코딩이~

▶ **These are Basic Python!**

You have learned
90% of Python!

▶ Advanced Python Topics

Simple Format String

List Generating

Generator (yield)

Lambda Function (lambda)

► Simple Format String

```
print "You gave me " + count + " apples"
```

```
print "He gave me " + count + " apples " + ...
```



```
print "You gave me %d apples" % count
```

▶ Simple Format String

Format String	형식
%d	Integer
%s	String
%f	Float

```
print "%5d" % 30
```

```
print "%06.4f" % 0.984578627387
```

▶ List Generating

실습 #1 !

```
list1 = range(50)
```

```
list2 = range(100, 150)
```

```
[(0, 100), (0, 101), ... ,(49, 100), ... (49, 149)]
```

실습 #2 !

```
list3 = range(50)
```

```
[(0, 0), (1, 1), (2, 8), (3, 27) ... (x, x **3) ...]
```

▶ List Generating

```
ans1 = [(x, y) for x in list1 for y in list 2]
```

```
ans2 = [(x, x**3) for x in list3]
```

{ (x, y) | x는 list1의 원소, y는 list2의 원소 }

{ (x, x³) | x는 list3의 원소 }

▶ List Generating

```
text = [ '    dirty', ' tExt  ', 'coLLEcTion  ' ]
```

```
text = [ x.strip().lower() for x in text ]
```

```
text = [ 'dirty', 'text', 'collection' ]
```

Python list [All hashable Objects!,]

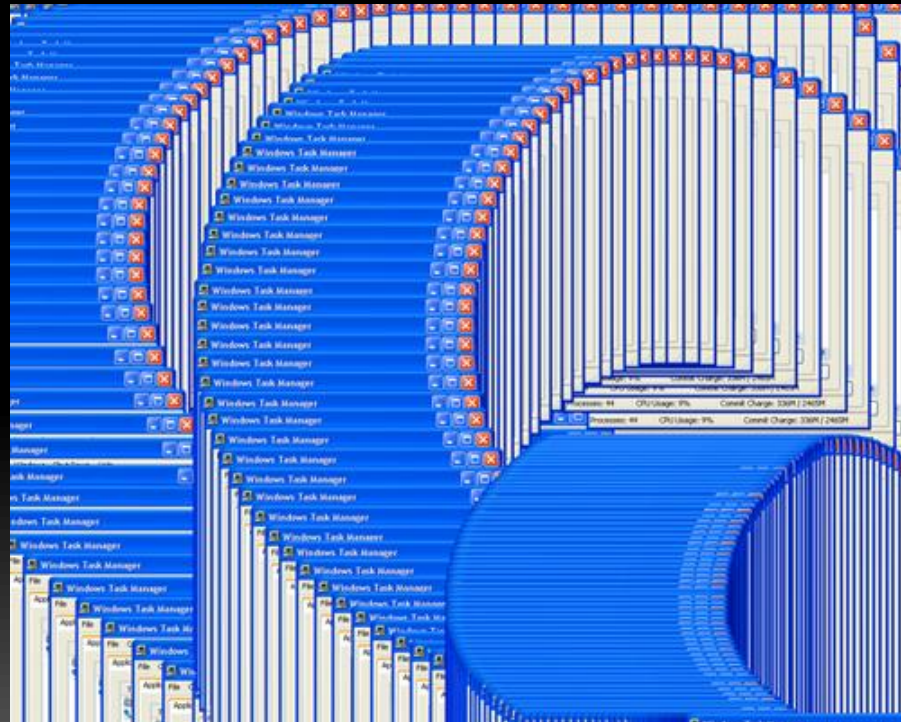
Function, Class, Tuple, Dictionary ...

무궁무진한 활용방법!

▶ Generator

```
mylist = range(100000000)
```

```
calc_result = [ x for x in 수천만개의 데이터 ]
```



▶ Generator

기본 아이디어

필요할 때마다 하나씩 주면 안되나?

From python 2.5!

Generator! 필요할때마다 하나씩!

yield

► Generator

```
1 #!/usr/bin/env python
2
3 # Let's think about huge list generation
4 #   mylist = range(1024*1024*1024)
5 # My list has size about 1G!!
6 # Can your computer handle this??
7
8 # To prevent this, we use generator, "yield"
9 def listmaker(range):
10     current = 0
11     while current < range:
12         yield current
13         current += 1
14
15 for x in listmaker(1024*1024*1024):
16     print x,
```

Use yield with for!

▶ “for” revisited – 약간 어려운 이야기

```
1 #!/usr/bin/env python
2
3 # This is simple for loop
4 for i in range(10):
5     print i
6
7 # Actually above for works like below
8 for i in [0,1,2,3,4,5,6,7,8,9]:
9     print i
10
11 # This is new range function, xrange
12 for i in xrange(10):
13     print i
14
15 # This gives us iterator, the generator
16 _x = xrange(10)
17 _iter = _x.__iter__()
18
19 # We can write above for like this
20 while True:
21     try:
22         i = _iter.next()
23         print i
24     except StopIteration:
25         # This exception is thrown(raised)
26         # when iteration is over
27         break
```

Python의 for는 사실..

C, C++

Java

등등의 for와는 다르다!

▶ Lambda Function

Lambda Function = 계산의 수학적 정의
= 계산을 함수화 한 정의

$$f(x) = x + 3$$

$$\lambda x : x + 3$$

$$f(x, y) = x * y$$

$$\lambda x, y : x * y$$

► Lambda Function

```
29 def absolute(is_minus, value):
30     if is_minus:
31         return value * -1
32     else:
33         return value
34
35 old_value = -3
36 is_minus = old_value < -1
37 new_value = absolute(is_minus, old_value)
38 print "abs(%s) = %s" % (old_value, new_value)
```



```
40 is_minus = old_value < -1
41 abs = is_minus and (lambda x : x * -1)
42 print "abs(%s) = %s" % (old_value, abs(old_value))
```

► You have learned

Function

Class with OOP

(Inheritance, Overloading, Overriding ...)

Package

Advanced Python Topics

Simple Format String

List Generating

Generator (yield)

Lambda Function (lambda)

▶ **Congratulations!**

You learned

99%

of python

► Congratulations!

Program with Python!

Think with Python!

Explore other

1%

Then you can master Python

▶ Homework

- You should review all the materials we have discussed
- You can get the codes used in this ppt
http://pipoket.kaist.ac.kr/sp_seminar/week02.tar.gz

REVIEW ALL THE SAMPLE CODES!